IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 4, 2019, accepted December 17, 2019, date of publication December 26, 2019,
date of current version January 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962600

Secure Data Sharing and Search for Cloud-Edge-Collaborative Storage

YE TAO[1,2], PENG XU[1,2], AND HAI JIN[1], (Fellow, IEEE)

[1]Services Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, National Engineering Research Center for Big Data Technology and System, Big Data Security Engineering Research Center, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China

Corresponding author: Hai Jin (hjin@mail.hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872412, in part by the Guangdong Provincial Science and Technology Plan Project under Grant 2017B010124001, in part by the Guangdong Provincial Key Research and Development Plan Project under Grant 2019B010139001, and in part by the Shenzhen Fundamental Research Program under Grant JCYJ20170413114215614.

ABSTRACT Cloud-edge-collaborative storage (CECS) is a promising framework to process data of the internet of things (IoT). It allows edge servers to process IoT data in real-time and stores them on a cloud server. Hence, it can rapidly respond to the requests of IoT devices, provide a massive volume of cloud storage for IoT data, and conveniently share IoT data with users. However, due to the vulnerability of edge and cloud servers, CECS suffers from the risk of data leakage. Existing secure CECS schemes are secure only if all edge servers are trusted. In other words, if any edge server is compromised, all cloud data (generated by IoT devices) will be leaked. Additionally, it is costly to request expected data from the cloud, which is linear with respect to the number of edge servers. To address the above problems, we propose a new secure data search and sharing scheme for CECS. Our scheme improves the existing secure CECS scheme in the following two ways. First, it enables users to generate a public-and-private key pair and manage private keys by themselves. In contrast, the existing solution requires edge servers to manage users' private keys. Second, it uses searchable public-key encryption to achieve more secure, efficient, and flexible data searching. In terms of security, our scheme ensures the confidentiality of cloud data and secure data sharing and searching and avoids a single point of breakthrough. In terms of performance, the experimental results show that our scheme significantly reduces users' computing costs by delegating most of the cryptographic operations to edge servers. Especially, our scheme reduces the computing and communication overhead for generating a search trapdoor compared with the existing secure CECS scheme.

INDEX TERMS Cloud-edge-collaborative storage, data sharing, data search, searchable encryption.

I. INTRODUCTION

Cloud-edge-collaborative storage (CECS) serves to equip edge servers between internet of things (IoT) devices and cloud servers. In CECS, it is clear that the cloud server is rich in storage and computing resources, edge servers are near to IoT devices, and IoT devices usually have limited resources. Hence, the edge servers rapidly respond to requests from IoT devices, e.g., analyzing data collected from IoT devices in real-time and forwarding processed data to the cloud server to save the cost of IoT devices. The authorized users can also conveniently share IoT data stored on the cloud server with the help of edge servers. Figure 1 shows the scenario of CECS. In it, IoT devices first collect and upload data to nearby edges. Second, edges process IoT data in real-time, return the result and store IoT data on a cloud server. Finally, users can share expected IoT data on the cloud server by submitting corresponding search requests.

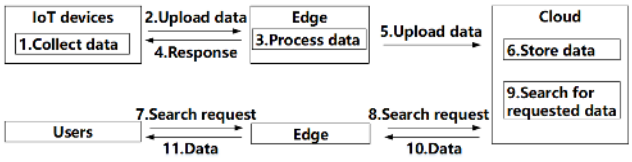

FIGURE 1. Scenario of CECS.

To retain data confidentiality in CECS, Mollah *et al.* [1] proposed a scheme in 2017 (called MA'17 in this paper) that

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wang.

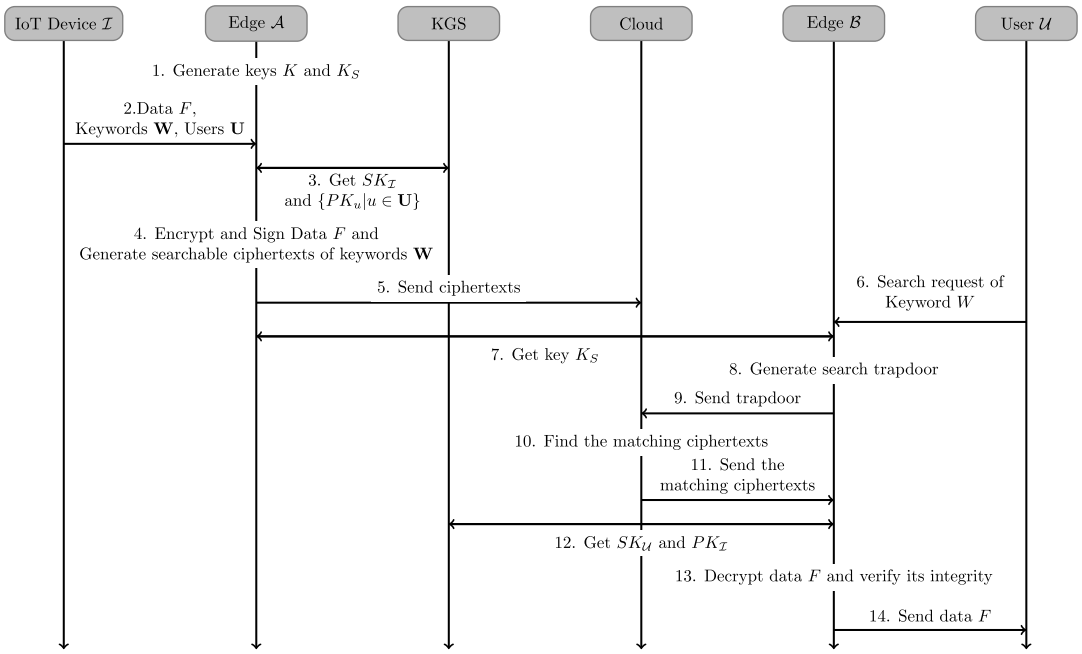

FIGURE 2. MA'17 scheme.

can securely share and search data for the cloud-and-edge-assisted IoT. MA'17 supposes that all edge servers are trusted and that the cloud server is honest-but-curious. Under these two assumptions, MA'17 consists of the following phases, as shown in Figure 2.

Setup phase: For each IoT device, the nearby edge server generates a data-sharing secret key K and a data-search secret key K_S.

Data Uploading phase. When an IoT device (denoted by $\mathcal{I}$) wants to store collected data F on the cloud server, device $\mathcal{I}$ sends data F, the extracted keywords $\mathbf{W}$, and the authorized users list $\mathbf{U}$ to a nearby edge server (denoted by $\mathcal{A}$). Edge $\mathcal{A}$ retrieves the private key $SK_{\mathcal{I}}$ of device $\mathcal{I}$ and the public keys $\{PK_u|u \in \mathbf{U}\}$ of the authorized users from the key generation center (KGC). It then encrypts data F with secret key K, generates searchable symmetric-key ciphertexts with secret key K_S and keywords $\mathbf{W}$, and encrypts secret key K with public keys $\{PK_u|u \in \mathbf{U}\}$. Finally, it signs data F with private key $SK_{\mathcal{I}}$ to ensure the data integrity and uploads the above-generated ciphertexts and signature to the cloud server.

Data Search phase. To share the expected data of device $\mathcal{I}$ via the cloud server, an authorized user (denoted by $\mathcal{U}$, where $\mathcal{U} \in \mathbf{U}$) chooses a keyword W as his search request and sends the request to a nearby edge server (denoted by $\mathcal{B}$). Edge $\mathcal{B}$ retrieves secret key K_S of edge $\mathcal{A}$, generates the search trapdoor with keyword W and secret key K_S, and sends this trapdoor to the cloud server. The cloud server searches for matched ciphertexts and returns them to edge $\mathcal{B}$. Edge $\mathcal{B}$ retrieves the public key $PK_{\mathcal{I}}$ of device $\mathcal{I}$ from the KGC and the private key $SK_{\mathcal{U}}$ of user $\mathcal{U}$, decrypts the matched data F, verifies the integrity of data F with the public key $PK_{\mathcal{I}}$ of device $\mathcal{I}$, and finally sends data F to user $\mathcal{U}$.

Data Sharing phase. To share all data of device $\mathcal{I}$, user $\mathcal{U}$ sends his sharing request to edge $\mathcal{B}$. Edge $\mathcal{B}$ downloads the corresponding ciphertexts, decrypts all data, verifies the data integrity, and finally returns all data to user $\mathcal{U}$.

MA'17 is advantageous in reducing the cost of IoT devices by delegating cryptographic operations to edge servers. However, it still has the following problems in practice. First, according to steps 2 and 12 in Figure 2, edge servers can know the private keys of IoT devices. Thus, if an edge server is compromised, it can be used to forge IoT data. Second, according to step 7 in Figure 2, edge servers trust each other and share their data-search secret keys. Thus, a compromised edge server can be used to generate search trapdoors with arbitrary keywords and retrieve expected data from the cloud server. Finally, suppose that a (mobile) IoT device can upload its data via different and uncertain edge servers. Step 4 in Figure 2 implies that the nearby edge server must fetch the data-search secret keys of all edge servers to retrieve the expected data for an authorized user. The communication cost is vast if there are many edge servers in practice.

A. CONTRIBUTIONS

To address the above problems, we propose a new CECS scheme. First, the new scheme allows IoT devices and users to generate their public-and-private keys by themselves, and all private keys are thus known only by their generators. Hence, the new scheme achieves more secure private-key management and resists data forgery. Second, the new scheme applies searchable public-key encryption instead of searchable symmetric-key encryption. Thus, users can generate keyword search trapdoors with their private keys and keywords, and a compromised edge server cannot be used to generate

keyword search trapdoors. Moreover, the nearby edge server of a user does not need to fetch the secret keys of all edge servers. Additionally, the new scheme also saves the cost of IoT devices and users by delegating cryptographic operations to the nearby edge servers as much as possible. Finally, we show that the new scheme is more secure than MA'17 and experimentally test its performance.

B. RELATED WORKS ON SECURE CLOUD AND EDGE COMPUTING

Edge computing is flourishing with the rapid development of cloud computing as a new computing paradigm. Consequently, data confidentiality in the cloud and edge computing are capturing increasing attention from academia and industries [2], [3].

1) CLOUD SECURITY

Numerous researchers have used various cryptographic schemes to achieve data security in cloud storage [4]. Xu *et al.* introduced a general hybrid proxy re-encryption (PRE) scheme [5] in cloud storage to protect data security against a curious cloud while achieving secure data sharing. Then, Zeng *et al.* introduced a conditional PRE scheme [6] to achieve fine-grained data sharing delegation. Attribute-based encryption was introduced into cloud computing to achieve fine-grained access control over outsourced encrypted data [7]–[10]. The problem of how to enable a semi-trusted cloud to compute between ciphertexts while guaranteeing the privacy of the encrypted data has also attracted significant attention [11], [12]. As a result, homomorphic encryption was introduced [13]–[15].

How to conduct a secure search over encrypted data is also a hot topic. For this purpose, searchable encryption has been proposed. Song *et al.* [16] first introduced a searchable symmetric encryption (SSE) scheme. To make SSE more practical, Kamara *et al.* [17] introduced dynamic searchable symmetric encryption (DSSE). Afterward, Xu *et al.* [18] improved the practicability of DSSE and enhanced the security. Recently, Ghareh Chamani *et al.* [19] proposed a new construction for forward and backward private DSSE, which reduces leakage of the SSE.

Boneh *et al.* [20] first introduced a public-key encryption with keyword search (PEKS) scheme for a single keyword search. Following the first study on PEKS, some researchers devoted efforts to making PEKS versatile. For example, Shi *et al.* [21] proposed a multi-dimensional range query scheme on ciphertexts, and Boneh and Waters [22] introduced a PEKS scheme supporting range, subset, and conjunctive queries. However, the PEKS schemes were faced with the obstacle of low retrieval efficiency with linear retrieval complexity. In 2015, Xu *et al.* [23] introduced the structured PEKS scheme and first achieved sub-linear retrieval complexity. Subsequently, Xu *et al.* devoted effort to researching the application of PEKS in IoT scenarios and proposed a parallel keyword search scheme [24] for the cloud-assisted IoT and a lightweight PEKS [25] scheme for cloud-assisted wireless sensor networks. In addition to confidentiality, remote data integrity [26] is another concern for secure outsourcing storage. Accordingly, proof of data possession [27], [28] and proof of data retrievability [29], [30] were proposed to verify the cloud data integrity.

2) EDGE SECURITY

Researchers are committed to various aspects of security issues, such as trusted devices, access control, network security, and intrusion detection [3]. Pettersen *et al.* [31] attempted to build a prototype utilizing secure enclave technologies on edge devices to enforce security isolation. Vassilakis [32] utilized a formal methodology to deploy policy enforcement components in mobile edge computing. To achieve secure communication in edge environments, Pimentel *et al.* [33] proposed a secure communication protocol for federated content networks. Chen *et al.* [34] introduced a deep-learning-based model in mobile edge computing to detect malicious applications at the cellular network edge.

3) CLOUD-EDGE-COLLABORATIVE SECURITY

All of the above works only take the security of either cloud storage or edge storage into consideration. In other words, there is almost no research on the security of cloud-edge-collaborative storage. One of the closest works was proposed by Mollah *et al.* [1] to protect the data privacy of outsourced storage in the the cloud-and-edge-assisted IoT. This work demonstrated that by deploying searchable encryption (SE) along with another cryptographic algorithm, it was possible to share and search outsourced data with privacy preservation in the cloud-edge-collaborative model. However, the scheme in [1] is not secure enough, as edges can obtain all private keys of mobile objects. All edges share the data-search secret key, which easily leads to the problem that any edge server could be compromised and then leveraged to break the security of the entire system.

C. ORGANIZATION

The remaining sections are organized as follows. Section II defines the cryptographic primitives used in this paper. Section III models the CECS system and clarifies its security goals. Section IV instantiates our CECS system. Section V analyzes the performance of the CECS system. Section VI concludes this paper.

II. BACKGROUND

Our CECS system utilizes symmetric encryption, public-key encryption, digital signature, and PEKS to realize the function and security goals. In this section, we briefly review these cryptographic primitives.

Symmetric encryption (SE) is a cryptographic primitive that encrypts data or decrypts ciphertexts with the same secret key. We define SE as follows:

Definition 1 (Symmetric Encryption): A symmetric encryption scheme **SE** *is composed of the following algorithms*:

- **SE.Setup**(1^k)*:Takes as input a security parameter* 1^k *and probabilistically outputs secret key* K_{SE}.
- **SE.Enc**(K_{SE}, M)*: Takes as inputs a plaintext* M *and secret key* K_{SE} *and probabilistically outputs a ciphertext* C_{SE}.
- **SE.Dec**(K_{SE}, C_{SE})*: Takes secret key* K_{SE} *and ciphertext* C_{SE} *as inputs and deterministically outputs plaintext* M.

For correction, an SE scheme requires that for any secret key K_{SE} and plaintext M (let C_{SE} be **SE.Enc**(K_{SE}, M)), $M =$ **SE.Dec**(K_{SE}, C_{SE}) holds except for a negligible probability. For security, an SE scheme requires that without the correct secret key, no probabilistic polynomial time (PPT) adversary can distinguish any two SE ciphertexts encrypted with the secret key K_{SE}.

Public key encryption (PKE) is a cryptographic primitive that has different encryption key and decryption key. Furthermore, users can publish the encryption key, and the decryption must be kept secret. PKE is defined as follows:

Definition 2 (Public Key Encryption): A public key encryption scheme **PKE** *consists of the following algorithms*:

- **PKE.Setup**(1^k)*: Takes as input security parameter* 1^k *and outputs a pair of public-and-private keys* (PK_{PKE}, SK_{PKE}).
- **PKE.Enc**(PK_{PKE}, M)*: Takes a public key* PK_{PKE} *and a plaintext* M *as inputs and probabilistically outputs a ciphertext* C_{PKE}.
- **PKE.Dec**(SK_{PKE}, C_{PKE})*: Takes a private key* SK_{PKE} *and a ciphertext* C_{PKE} *as inputs and outputs a plaintext* M.

For correction, a PKE scheme requires that for any key pair (PK_{PKE}, SK_{PKE}) and plaintext M (let C_{PKE} be **PKE.Enc**(PK_{PKE}, M)), $M =$ **PKE.Dec**(SK_{PKE}, C_{PKE}) holds except for a negligible probability. For security, a PKE scheme requires that without the correct SK_{PKE}, a PPT adversary cannot distinguish any two PKE ciphertexts encrypted with the public key PK_{PKE}.

A digital signature (DS) is a cryptographic primitive for identifying digital information based on public-key encryption technology. The signer uses his private key to sign a message and publishes his public key; the verifier can prove that the signature belongs to the signer with the signer's public key. We define DS as follows:

Definition 3 (Digital Signature): A digital signature scheme **DS** *contains the following algorithms*:

- **DS.Setup**(1^k)*: Takes as input security parameter* 1^k *and then outputs a pair of public-and-private keys* (PK_{DS}, SK_{DS}).
- **DS.Sig**(SK_{DS}, M)*: Takes as inputs a message* M *and a private key* SK_{DS} *and then outputs a signature* Sig.
- **DS.Ver**(PK_{DS}, M, Sig)*: Takes a signature* Sig*, a message* M*, and a public key* PK_{DS} *as inputs and then outputs* 1 *if* Sig *is valid or* 0 *otherwise.*

The correction of a DS scheme requires that for any key pair (PK_{DS}, SK_{DS}) and message $M \in \{0,1\}^*$ (let Sig be **DS.Sig**(SK_{DS}, M)), $1 =$ **DS.Ver**(PK_{DS}, M, Sig) holds except for a negligible probability. For security, a DS scheme requires that without the private key, no one can forge a signature, except for a negligible probability.

PEKS enables secure keyword search over ciphertexts by setting the public key. It is defined as follows:

Definition 4 (Public Key Encryption with Keyword Search): A public key encryption with keyword search scheme **PEKS** consists of the following algorithms:

- **PEKS.Setup**(1^k): Takes as input a security parameter 1^k and outputs a public-and-private key pair (PK_{PEKS}, SK_{PEKS}).
- **PEKS.Enc**(PK_{PEKS}, W): Take as inputs a public key PK_{PEKS} and a keyword W and then probabilistically outputs a searchable ciphertext C_{PEKS} of keyword W.
- **PEKS.Trapdoor**(SK_{PEKS}, W): Takes a keyword W and a private key SK_{PEKS} as inputs and then outputs a keyword trapdoor T_W.
- **PEKS.Search**$(PK_{PEKS}, \mathbb{C}_{PEKS}, T_W)$: Takes a public key PK_{PEKS}, a sequence of ciphertexts $\mathbb{C}_{PEKS}$, and a trapdoor T_W as inputs and outputs the ciphertexts $\mathbb{C}_W \subseteq \mathbb{C}_{PEKS}$ of keyword W.

In terms of correction, a PEKS scheme requires that for any key pair (PK_{PEKS}, SK_{PEKS}) and keyword W, **Search**$(PK_{PEKS}, \mathbb{C}_{PEKS}, T_W)$ (where T_W is generated by **Trapdoor**(SK_{PEKS}, W)) outputs all ciphertexts of keyword W except for a negligible probability. For security, a PEKS scheme requires that without SK_{PEKS}, a PPT adversary cannot distinguish any two PEKS ciphertexts of public key PK_{PEKS}.

III. CLOUD-EDGE-COLLABORATIVE STORAGE MODEL

In this section, we model our CECS scheme and clarify its security goals.

A. OBJECTS

Our CECS scheme has five entities: IoT devices, users, edges, cloud, and certificate authority (CA).

- IoT devices can store their data on the cloud through nearby edges.
- Users can download or retrieve the data shared by IoT devices from the cloud through nearby edges.
- Edges encrypt data or decrypt ciphertexts for IoT devices and users and communicate with the cloud.
- The cloud is responsible for storing ciphertexts generated by edges and returning the data that the edges request.
- The CA issues digital certificates of IoT devices and users.

B. FUNCTIONS

The workflow of our CECS scheme is divided into 4 phases, as follows:

- **Setup** phase: IoT devices and users register at the CA; then, the CA stores their certificates.
- **Data Uploading** phase: An IoT device sends its data and the corresponding keywords to a nearby edge. The edge encrypts the data and stores ciphertexts on the cloud.

- **Data Sharing** phase: An authorized user submits a sharing request to a nearby edge. The edge requests and downloads the encrypted data of the requested IoT devices from the cloud, decrypts them and verifies their integrity. Finally, the edge returns the decrypted data to the user.
- **Data Search** phase: An authorized user generates a search request to the cloud through a nearby edge. The cloud searches for the matched ciphertexts and sends them to the edge. The edge decrypts them and verifies their integrity. Finally, the edge returns the decrypted data to the user.

C. SECURITY ASSUMPTIONS AND GOALS

Our CECS scheme is built among IoT devices, users, edges, cloud, and the CA. Because these objects are different, we describe four trust models to clarify the role of each object in our scheme. The trust models are as follows:

- Trust Model 1: IoT devices and users trust their nearby edges. This model is reasonable since (1) an edge is deployed for its IoT devices and users and (2) compared with the cloud, the data at edges are more scattered and therefore less likely to be attack targets. This model is the same as MA'17.
- Trust Model 2: Any two edges do not directly trust each other. Any edge could be compromised, so we should prevent the compromised edge from being leveraged to break other edges' security. Additionally, different edges may be deployed by different companies, and those companies may compete. This model is different from MA'17, as MA'17 assumes that edges trust each other.
- Trust Model 3: The cloud is generally assumed to be honest-but-curious, which means that the cloud will provide services honestly, but the cloud remains curious about users' data. This model is the same as MA'17.
- Trust Model 4: Any outside attacker is untrusted. This model is a usual one and the same as MA'17.

Our CECS model achieves the following security goals:

- Maintaining the confidentiality of the IoT device data stored on the cloud. All data must be secret to the cloud and outside attackers. This goal is the same as that of MA'17.
- Securely sharing data. This goal requires that only authorized users are allowed to download the shared data and verify the data integrity through edges. This security is the same as that of MA'17.
- Securely retrieving data. This goal requires that only the authorized users are allowed to retrieve the shared data, and no important characteristic information about the shared data is leaked to the cloud. This goal is the same as that of MA'17.
- Avoiding a single point of breakthrough. If an outside attacker attacks an edge, it cannot undermine the security of other edges. This security is a new goal compared with MA'17.

IV. CLOUD-EDGE-COLLABORATIVE STORAGE SYSTEM

In this section, we instantiate our CECS scheme. Our scheme consists of the following four phases.

In the **Setup** phase, each of the IoT devices and users generates a public-and-private key pair and then registers the public key to the certificate authority (CA) while keeping the private key secret. Compared with MA'17, our scheme is more secure since all IoT devices and users manage their private keys by themselves.

The **Setup** phase is shown in Figure 3. For each IoT device or user, the details are as follows:

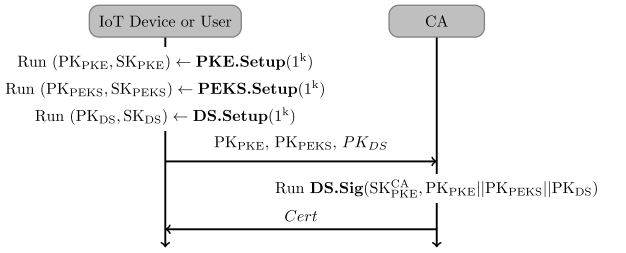

FIGURE 3. Setup phase of our CECS scheme.

1) The IoT device or user runs algorithms $(PK_{PKE}, SK_{PKE}) \leftarrow \mathbf{PKE.Setup}(1^k)$, $(PK_{PEKS}, SK_{PEKS}) \leftarrow \mathbf{PEKS.Setup}(1^k)$, and $(PK_{DS}, SK_{DS}) \leftarrow \mathbf{DS.Setup}(1^k)$.
2) The device or user registers its/his public keys PK_{PKE}, PK_{PEKS}, and PK_{DS} to the CA.
3) The CA runs algorithm $\mathbf{DS.Sig}(SK^{CA}_{PKE}, PK_{PKE}||PK_{PEKS}||PK_{DS})$ to generate the certificate $Cert$.
4) The CA sends the certificates $Cert$ to the IoT device or user.

In the **Data Uploading** phase, an IoT device signs its data with its private key. It sends the data, the corresponding signature, the extracted keywords from the data, its certificate, and a list of the authorized users to a nearby edge server via a secure channel. Then, the edge server generates a secret key K and encrypts the received data with this secret key. To securely share data, the edge server encrypts the secret key with the public key of the authorized users. Although using broadcast encryption can reduce the communication and storage costs of the cloud, an authorized user needs to acquire other authorized users' public keys for decrypting a broadcast ciphertext, which will increase the communication and computing costs of the user. Hence, we still use the conventional public-key encryption in the above. To support secure data retrieval, we apply PEKS to generate keyword searchable ciphertexts with the authorized users' public keys. Compared with MA'17, our scheme reduces the communication and computing costs of edge servers in making secure search queries for users.

The **Data Uploading** phase is shown in Figure 4. The details are as follows:

1) An IoT device logs into a nearby edge server, extracts keywords $\mathbf{W}$ from its collected data F, and runs algorithm $Sig \leftarrow \mathbf{DS.Sig}(SK^O_{DS}, F)$ to generate data signature Sig with its private key SK^O_{DS}.

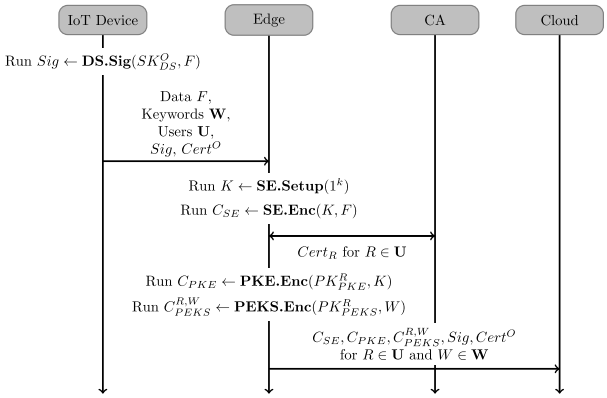

FIGURE 4. Data uploading phase of our CECS scheme.

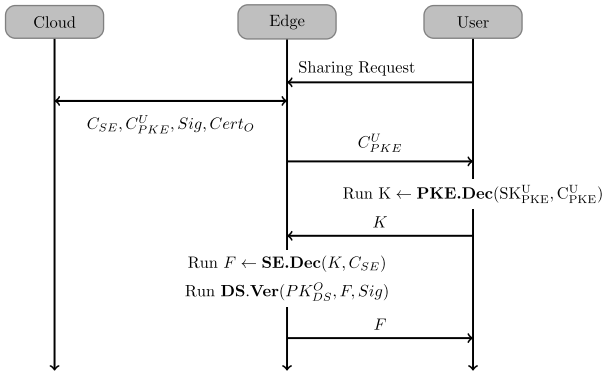

FIGURE 5. Data sharing phase of our CECS scheme.

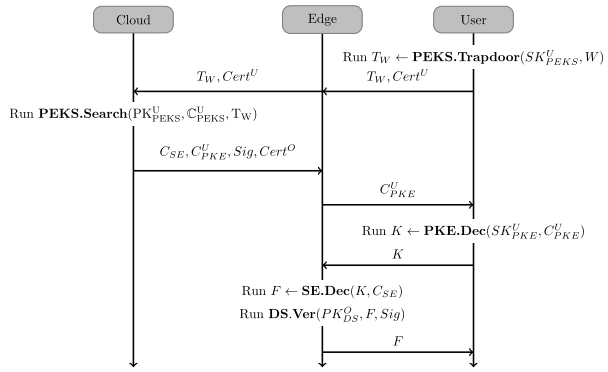

FIGURE 6. Data search phase of our CECS scheme.

2) The IoT device uses a secure channel to send data F, keywords **W**, signature Sig, its certificate $Cert^O$, and a list **U** of authorized users to the edge server.
3) The edge server runs algorithms $K \leftarrow$ **SE.Setup**(1^k) and $C_{SE} \leftarrow$ **SE.Enc**(K, F) to encrypt data F in the setting of the symmetric key.
4) The edge server fetches the authorized users' certificates $\{Cert^R | R \in \mathbf{U}\}$ from the CA and obtains the authorized users' public keys $\{PK^R_{PKE}, PK^R_{PEKS}, PK^R_{DS} | R \in \mathbf{U}\}$.
5) The edge server runs algorithms

$$C^R_{PKE} \leftarrow \mathbf{PKE.Enc}(PK^R_{PKE}, K)$$

and

$$C^{R,W}_{PEKS} \leftarrow \mathbf{PEKS.Enc}(PK^R_{PEKS}, W)$$

for $R \in \mathbf{U}$ and $W \in \mathbf{W}$.
6) The edge server uploads ciphertexts $C_{SE}, C^R_{PKE}, C^{R,W}_{PEKS}, Sig, Cert^O$ for $R \in \mathbf{U}$ and $W \in \mathbf{W}$ to the cloud.
7) The cloud stores the received ciphertexts.

In the **Data Sharing** phase, an authorized user submits a data sharing request to the nearby edge server. The edge server requests and obtains the corresponding ciphertexts from the cloud and sends the contained PKE ciphertext to the authorized user. Next, the authorized user decrypts the secret key and uses a secure channel to send this key to the edge server. After receiving the secret key, the edge decrypts data and uses a secure channel to return the data to the authorized user if the data signature is valid.

The **Data Sharing** phase is shown in Figure 5. To share the data of IoT device O, the details are as follows:

1) An authorized user (denoted by U) submits a data sharing request to the nearby edge server.
2) The edge server requests and downloads ciphertexts $C_{SE}, C^U_{PKE}, Sig, Cert^O$ of device O from the cloud.
3) The edge server acquires the public keys PK^O_{PKE} and PK^O_{DS} from certificate $Cert^O$.
4) The edge sends ciphertext C^U_{PKE} to the authorized user U.
5) The authorized user U runs algorithm $K \leftarrow$ **PKE.Dec**(SK^U_{PKE}, C^U_{PKE}) and uses a secure channel to return secret key K to the edge server.
6) The edge server runs algorithm $F \leftarrow$ **SE.Dec**(K, C_{SE}), verifies the validity of data F by running algorithm **DS.Ver**(PK^O_{DS}, F, Sig), and returns data F to the users via a secure channel if data F are valid.

In the **Data Search** phase, an authorized user generates a keyword search trapdoor with his private key and desired keyword and submits this trapdoor and his certificate to the cloud via a nearby edge server. The cloud searches for the matched ciphertexts and sends them to the edge server. The edge server decrypts the requested data as it does in the **Data Sharing** phase and returns the data to the user via a secure channel if the data are valid.

The **Data Search** phase is shown in Figure 6. To share the data with the expected keyword W from IoT device O, the details are as follows:

1) An authorized user (denoted by U) runs algorithm $T_W \leftarrow$ **PEKS.Trapdoor**(SK^U_{PEKS}, W) and submits trapdoor T_W and his certificate $Cert^U$ to the nearby edge server.
2) The edge server forwards the received request to the cloud.
3) The cloud obtains the public key PK^U_{PEKS} of the authorized user from certificate $Cert^U$.
4) The cloud runs algorithm **PEKS.Search**$(PK^U_{PEKS}, \mathbb{C}^U_{PEKS}, T_W)$ to find the matching ciphertexts, where

TABLE 1. Comparisons of communication costs.

Scheme	Phase	Communication Trips				
		IoT Device	Edge (near to the IoT device)	Cloud	Edge (near to the user)	User
Our Scheme	**Data Uploading**	1	4	3	0	0
	Data Sharing	0	0	2	8	4
	Data Search	0	0	4	8	4
MA'17	**Data Uploading**	1	4	1	0	0
	Data Sharing	0	0	2	6	2
	Data Search	0	0	2	$6 + 2 * \lvert Edges \rvert$	2

Note: $\lvert Edges \rvert$ denotes the number of remaining edge servers.

$\mathbb{C}^{U}_{PEKS}$ denotes all PEKS ciphertexts generated by device O for user U.

5) Supposing ciphertext $C_{SE}, C^{U}_{PKE}, Sig, Cert^{O}$ to be a matching one, the cloud returns this ciphertext to the edge server.
6) The edge server and the user do the same as in the **Data Sharing** phase; finally, the user receives the data of the keyword W.

A. SECURITY ANALYSIS

Our CECS scheme achieves the following three security goals: data confidentiality on the cloud, secure data sharing, and secure data search. Moreover, compared with MA'17, our scheme is more secure.

First, we analyze the confidentiality of data on the cloud. In our CECS scheme, edge servers encrypt IoT devices' data locally before uploading them to the cloud. The cloud stores the SE ciphertexts, the PKE ciphertexts, and the PEKS ciphertexts. According to the security requirements of SE, PKE, and PEKS, no one without secure or private keys can infer the content of IoT device data from the above ciphertexts. Hence, all data are secret to the cloud and outside attackers.

Next, we analyze secure data sharing. In our CECS scheme, the shared data secret key is encrypted with authorized users' public keys. According to the security of PKE, only the authorized users can access the shared data through the nearby edge server, and it is not possible to decrypt any data that does not belong to the IoT device or is not shared with the requested user. According to the security of the digital signature, users can verify the IoT device's signature of the shared data through a nearby edge to ensure data integrity.

Then, we analyze the security of the data search. It is easy to find that our search solution has the same essence in terms of security as a PEKS scheme. According to the security of PEKS, no important characteristic information about the requested data is leaked to the cloud; moreover, outside attackers cannot perform the search.

Finally, we analyze avoiding a single point of breakthrough. If an outside attacker compromises an edge server, because of the security of DS, it cannot forge shared data without the IoT device's private key. According to the security of SE and PKE, the outside attacker cannot decrypt other ciphertexts on the cloud without the correct symmetric keys and private keys of the authorized users. Therefore, it cannot undermine the security of additional data on the cloud.

Compared with MA'17, in terms of key management, our scheme lets IoT devices and users generate public-and-private keys and manage private keys by themselves. However, in MA'17, the users' public-and-private keys are generated by the KGC, and edge servers can request users' private keys from the KGC. Thus, if an attacker compromises an edge server, it can use an authorized user's private key to decrypt the shared data and forge IoT device data with this device's private key.

In terms of data search, our scheme uses PEKS, and the authorized users can generate trapdoors with their private keys and submit these trapdoors to edge servers. However, the MA'17 scheme uses SSE, and edge servers share the data-search secret keys. Thus, if an attacker compromises an edge server, it can generate trapdoors with arbitrary keywords and submit these trapdoors to the cloud to find the matching data.

B. PERFORMANCE ANALYSIS

Table 1 shows a comparison of the communication cost between our CECS scheme and MA'17.

In the **Data Uploading** phase, our scheme requires the IoT device to generate the data signature and upload the data, the extracted keywords, the authorized users, the generated signature, and its certificate to the nearby edge server. Therefore, compared with MA'17, our scheme requires additional computing and communication overhead to upload data, but our scheme enables stronger security.

In the **Data Sharing** phase, our scheme requires the edge server to send the shared PKE ciphertext to the authorized user. Next, the authorized user decrypts the received ciphertext with his private key and returns a data-sharing secret key to the edge server. Therefore, compared with MA'17, our scheme requires additional computing and communication overhead when sharing data, but our scheme enables stronger security.

In the **Data Search** phase of our scheme, the authorized user only needs to generate one keyword search trapdoor. However, in MA'17, the requested edge server fetches all data-search secret keys from all of the other edge servers, creates keyword search trapdoors with these received data-search secret keys, and submits these trapdoors to the cloud. In other words, the computing and communication costs of the edge server in MA'17 are linear with respect to the number of edge servers. Therefore, our scheme reduces

TABLE 2. Configuration of System Parameters.

	Edge Server and Cloud Server	IoT Device
CPU	Intel(R) Core(TM) i5-3210M 2.5 GHz	Hisilicon Kirin 980 2*A76 2.6 GHz+2*A76 1.92 GHz+4xA55 1.8 GHz
RAM	8 GB	6 GB
Operating System	Windows 10 Professional 64 bit	Android 9
Runtime Environment	Jre1.8.0_201	Jre1.8.0_201

the communication and computing overhead for generating keyword search trapdoors.

V. EXPERIMENTS

In this section, we experimentally evaluate our CECS scheme's performance. Table 2 shows the software and hardware environments. We apply SHA-256, RSA-1024, and AES-128 to respectively implement a hash function, public-key encryption, and symmetric encryption. We utilize the PEKS scheme proposed in [20] to perform a keyword search over public-key ciphertexts. We code the above cryptographic primitives using JPBC (Java Pairing-Based Cryptography Library) [35] to implement PEKS. Table 3 shows the bilinear pairing parameters.

TABLE 3. Bilinear mapping parameters.

Elliptic Curve	$y^2 = x^2 + x$
Base Field	878071079966331252243778198475404981580688319941420821102865339926647563088022295707862517942266222142315585876958231745927771336731748132492512999822479 1
Group Order	730750818665451621361119245571504901405976559617

A. EVALUATING THE DATA UPLOADING PHASE

To test the performance of our CECS scheme in the **Data Uploading** phase, we upload the following 4 data sets to the cloud server: 1 MB of data containing 21 keywords, 5 MB of data containing 105 keywords, 10 MB of data containing 205 keywords, and 50 MB of data containing 1,024 keywords. Figure 7 shows our CECS scheme's time cost in the **Data Uploading** phase. For example, when an IoT device uploads 50 MB of data to the cloud, the time for the device to sign the data is approximately 840 ms, the total time for the nearby edge server to generate the corresponding ciphertexts is approximately 46,970 ms, and the total time of the **Data Uploading** phase is approximately 60,863 ms. When the data size expands, the time cost for the user to upload data also increases, but the time cost of the IoT device is very little compared with the total time cost.

B. EVALUATING THE DATA SHARING PHASE

To test the performance of our CECS scheme in the **Data Sharing** phase, we download the following 4 data sets from the cloud server: 1 MB of data containing 21 files, 5 MB of data containing 105 files, 10 MB of data containing 205 files, and 50 MB of data containing 1024 files. Figure 8 shows our CECS scheme's time cost in the **Data Sharing** phase. For example, when an authorized user requests 50 MB of shared data, the time for the user to decrypt secret keys is approximately 570 ms, the total time for the nearby edge server to decrypt data and verify the data integrity is approximately 916 ms, and the total time of the **Data Sharing** phase is approximately 13,648 ms. Therefore, when the data size expands, the time cost for the user to obtain shared data also increases, but the time cost for the user to decrypt secret keys is much less than the total time cost.

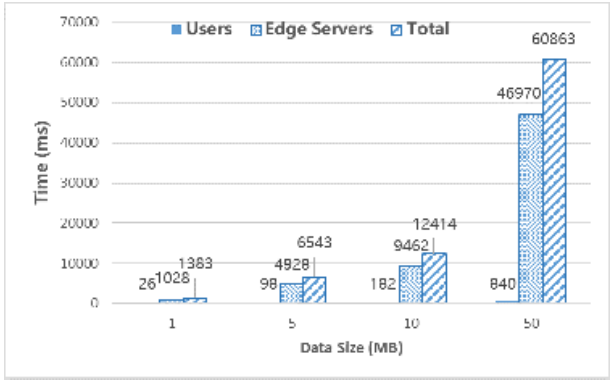

FIGURE 7. Time cost of our CECS scheme in the Data Uploading phase.

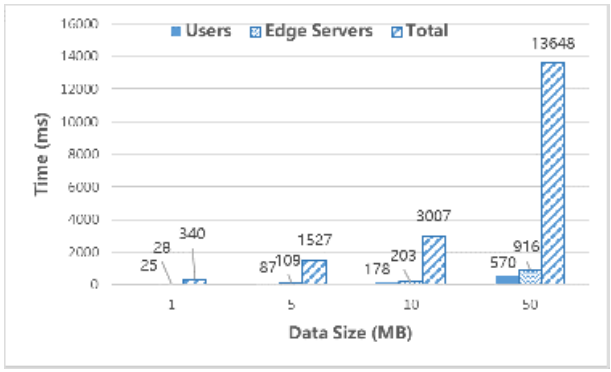

FIGURE 8. Time cost of our CECS scheme in the Data Sharing phase.

C. EVALUATING THE DATA SEARCH PHASE

To test the performance of our CECS scheme in the **Data Search** phase, we authorize the cloud to search keywords over different numbers of ciphertexts. The data sets are as follows: 1 MB of data containing 21 files, 5 MB of data containing 105 files, 10 MB of data containing 205 files,

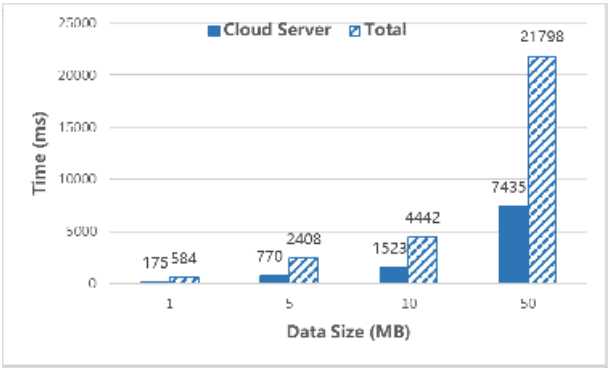

FIGURE 9. Time cost of our CECS scheme in the Data Search phase.

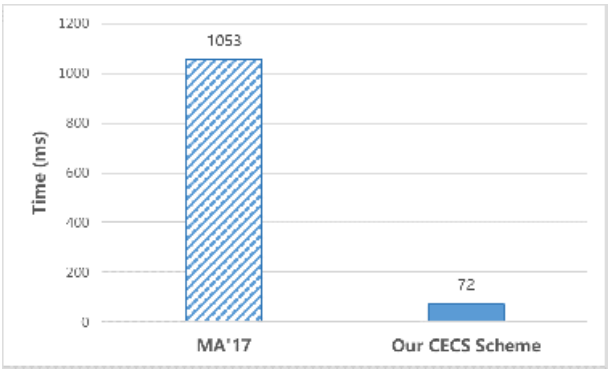

FIGURE 10. Time costs to generate keyword search trapdoors.

and 50 MB of data containing 1,024 files. Figure 9 shows our CECS scheme's time cost in the **Data Search** phase. For example, when the data size requested by the user is 50 MB, the time cost of the cloud to search over PEKS ciphertexts is approximately 7,435 ms, and the total time of the **Data Search** phase is approximately 21,798 ms.

D. PERFORMANCE IN GENERATING TRAPDOORS

To compare the performance of trapdoor generation between our CECS scheme and MA'17, we deploy ten edge servers. Figure 10 shows the time costs of trapdoor generation of our CECS scheme and MA'17. As shown in Figure 10, MA'17 requires approximately 1,053 ms for an edge server to obtain data-search secret keys from the other edge servers and generate ten trapdoors. However, in our CECS scheme, the authorized user only needs to create one trapdoor with the time cost of approximately 72 ms. Moreover, as the number of edge servers increases, the time cost of trapdoor generation of MA'17 also increases.

In terms of communication cost of trapdoor generation, the requirement for communication trips in MA'17 is linear with respect to the number of edge servers. However, in our CECS scheme, no communication trip is taken to generate a keyword search trapdoor. Therefore, our CECS scheme can significantly reduce the computing and communication overhead for creating a keyword search trapdoor.

VI. CONCLUSION

In this paper, we propose a new secure data search and sharing scheme for CECS. Compared with the previous work, our scheme is advantageous with respect to maintaining the privacy of IoT devices and users' private keys and achieving more secure or efficient data sharing and data searching. In terms of security, our scheme ensures the confidentiality of data on the cloud, secure data sharing between IoT devices and users and secure data searching between the cloud and users and enables the weak trust assumption on edge servers. In terms of performance, the experimental results show that our scheme also effectively reduces the computing burden of IoT devices and users by delegating computation-intensive encryption and decryption algorithms to edge servers. Compared with MA'17, our scheme significantly reduces the computing and communication overhead for generating keyword search trapdoors.

REFERENCES

[1] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud–assisted Internet of Things," *IEEE Cloud Comput.*, vol. 4, no. 1, pp. 34–42, Jan. 2017.
[2] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Commun. Surveys Tutr.*, vol. 15, no. 2, pp. 843–859, 2nd Quart., 2013.
[3] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.
[4] N. Kaaniche and M. Laurent, "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Comput. Commun.*, vol. 111, pp. 120–141, Oct. 2017.
[5] P. Xu, J. Xu, W. Wang, H. Jin, W. Susilo, and D. Zou, "Generally hybrid proxy re-encryption: A secure data sharing among cryptographic clouds," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, pp. 913–918, 2016.
[6] P. Zeng and K.-K.-R. Choo, "A new kind of conditional proxy re–encryption for secure cloud storage," *IEEE Access*, vol. 6, pp. 70017–70024, 2018.
[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, 2006, pp. 89–98.
[8] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive efficient and provably secure realization," in *Proc. 14th Int. Conf. Pract. Theory Public Key Cryptogr.*, vol. 6571, 2011, pp. 53–70.
[9] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.
[10] B. Chandrasekaran and R. Balakrishnan, "An efficient Tate pairing algorithm for a decentralized key–policy attribute based encryption scheme in cloud environments," *Cryptography*, vol. 2, no. 3, p. 14, Jul. 2018.
[11] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.
[12] K. Mallaiah and S. Ramachandram, "Applicability of homomorphic encryption and CryptDB in social and business applications: Securing data stored on the third party servers while processing through applications," *Int. J. Comput. Appl.*, vol. 100, no. 1, pp. 5–19, Aug. 2014.
[13] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proc. 3rd ACM Workshop Cloud Comput. Secur. Workshop (CCSW)*, 2011, pp. 113–124.
[14] B. Jiang and Y. Zhang, "Securely min and k-th min computations with fully homomorphic encryption," *Sci. China Inf. Sci.*, vol. 61, no. 5, May 2018, Art. no. 058103.
[15] T. Kim and H. Kim, "POSTER: Can we use biometric authentication on cloud?: Fingerprint authentication using homomorphic encryption," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2018, pp. 813–815.
[16] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, Nov. 2002, pp. 44–55.

[17] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 965–976.
[18] P. Xu, S. Liang, W. Wang, W. Susilo, Q. Wu, and H. Jin, "Dynamic searchable symmetric encryption with physical deletion and small leakage," in *Proc. 22nd Australas. Conf. Inf. Secur. Privacy*, vol. 10342, 2017, pp. 207–226.
[19] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "new constructions for forward and backward private symmetric searchable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2018, pp. 1038–1055.
[20] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, 2004, pp. 506–522.
[21] E. Shi, J. Bethencourt, T.-H.-H. Chan, D. Song, and A. Perrig, "Multi–dimensional range query over encrypted data," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 350–364.
[22] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptogr. Conf.*, 2007, pp. 535–554.
[23] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, and H. Jin, "Generating searchable public–key ciphertexts with hidden structures for fast keyword search," *IEEE Trans. Inf. Forensic Security*, vol. 10, no. 9, pp. 1993–2006, Sep. 2015.
[24] P. Xu, X. Tang, W. Wang, H. Jin, and L. T. Yang, "Fast and parallel keyword search over public–key ciphertexts for cloud–assisted IoT," *IEEE Access*, vol. 5, pp. 24775–24784, 2017.
[25] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public–key encryption for cloud–assisted wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3712–3723, Aug. 2018.
[26] M. Sookhak, "Remote data auditing in cloud computing environments: A survey taxonomy and open issues," *ACM Comput. Surv.*, vol. 47, no. 4, p. 65, 2015.
[27] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609.
[28] N. Kaaniche and M. Laurent, "SHoPS: Set homomorphic proof of data possession scheme in cloud storage applications," in *Proc. IEEE World Congr. Services*, Jun. 2015, pp. 143–150.
[29] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.
[30] C. B. Tan, M. H. A. Hijazi, Y. Lim, and A. Gani, "A survey on proof of retrievability for cloud data integrity and availability: Cloud storage state-of-the-art issues solutions and future trends," *J. Netw. Comput. Appl.*, vol. 110, pp. 75–86, 2017.
[31] R. Pettersen, H. D. Johansen, and D. Johansen, "Secure edge computing with ARM TrustZone," in *Proc. 2nd Int. Conf. Internet Things, Big Data Secur.*, 2017, pp. 102–109.
[32] V. Vassilakis, "Security analysis of mobile edge computing in virtualized small cell networks," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov.*, 2016, pp. 653–665.
[33] H. M. Pimentel, S. Kopp, M. A. Simplicio, Jr., R. M. Silveira, and G. Bressan, "OCP: A protocol for secure communication in federated content networks," *Comput. Commun.*, vol. 68, pp. 47–60, Sep. 2015.
[34] Y. Chen, Y. Zhang, and S. Maharjan, "Deep learning for secure mobile edge computing," 2017, *arXiv:1709.08025*. [Online]. Available: https://arxiv.org/abs/1709.08025
[35] A. D. Caro. *The Java Pairing Based Cryptography Library*. Accessed: Nov. 2, 2019. [Online]. Available: http://libeccio.di.unisa.it/projects/jpbc/

YE TAO received the B.E. degree in software engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2016, where he is currently pursuing the master's degree in computer architecture. His research interests include cloud security and cryptography.

PENG XU received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2010. He was an Associate Research Fellow of the University of Wollongong, Australia, where he is currently an Associate Professor. He held a postdoctoral position at the Huazhong University of Science and Technology, from 2010 to 2013. His research interest includes cryptography. He has authored over 30 research articles and two books. He was the PI in eight grants, including three NSF projects.

HAI JIN (Fellow, IEEE) received the Ph.D. degree in computer engineering from HUST, in 1994. He was with The University of Hong Kong, from 1998 to 2000, and a Visiting Scholar with the University of Southern California, from 1999 to 2000. He is the Chief Scientist of the National 973 Basic Research Program Project of Virtualization Technology of Computing System. He has coauthored 15 books. He has published over 400 research articles. He is a member of the ACM. In 1996, he was awarded the German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz, Germany. He was awarded the Excellent Youth Award from the National Science Foundation of China, in 2001.